VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

PROFILOVÁNÍ DAT POMOCÍ IPFIX MEDIÁTORU

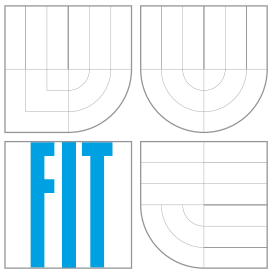
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MICHAL KOZUBÍK

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

PROFILOVÁNÍ DAT POMOCÍ IPFIX MEDIÁTORU

DATA PROFILING USING IPFIX MEDIATOR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MICHAL KOZUBÍK
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN KOŘENEK, Ph.D.
SUPERVISOR

BRNO 2015

Abstrakt

Tato práce se zabývá profilováním síťových dat pomocí IPFIX mediátoru. Hlavní úlohou je efektivní filtrování dat a konfigurovatelná správa profilů. Pro IPFIX mediátor dosud neexistuje řešení, které by tuto funkcionalitu zajišťovalo. Uživatelé tak ztrácí přehlednost při analýze nasbíraných dat a odhalování anomálií. Z toho důvodu je v bakalářské práci navrženo a implementováno řešení ve formě zásuvného modulu pro IPFIX mediátor, který využívá hierarchii profilů s definovanými filtračními pravidly pro třídění dat.

Abstract

This thesis deals with the network data profiling using IPFIX mediator. The main task is effective data filtering and configurable profiles management. The profiles management is still not available for IPFIX mediator, which makes analysis of network traffic for users more difficult. Therefore this thesis deals with the design and implementation of configurable profiles management as a plug-in for IPFIX mediator. The plug-in uses profiles hierarchy with filtering rules for data sorting.

Klíčová slova

IPFIX, monitorování, profilování, bezpečnost, klasifikace síťového provozu, propustnost

Keywords

IPFIX, monitoring, profiling, security, network traffic classification, throughput

Citace

Michal Kozubík: Profilování dat pomocí IPFIX mediátoru, bakalářská práce, Brno, FIT VUT v Brně, 2015

Profilování dat pomocí IPFIX mediátoru

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jana Kořenka, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Michal Kozubík
15. května 2015

Poděkování

Chtěl bych poděkovat vedoucímu své bakalářské práce Ing. Janu Kořenkovi, Ph.D. za vedení práce, odbornou pomoc a mnoho rad. Dále bych rád poděkoval Mgr. Petru Velanovi za odbornou spolupráci a řadu poskytnutých rad a informací.

Obsah

Seznam obrázků

Kapitola 1

Úvod

V poslední době jsme svědky rychlého rozvoje digitálních technologií. V roce 2005 překonal počet lidí připojených do sítě Internet hranici jedné miliardy. Dnes, tedy o 10 let později je toto číslo trojnásobné a dosahuje skoro poloviny celkové lidské populace na planetě. Z těchto čísel vyplývá, že celkové požadavky na tuto síť se stále zvyšují. Dříve lidem stačilo čtení primitivní webové prezentace či textového souboru. V dnešní době využívá Internet nespočet různých zařízení a služeb. Podíváme-li se kolem sebe, nalezneme spousty příkladů, počínaje těmi největšími, jakožto stolní (či sálové) počítače a laptopy, přes mobilní zařízení, kterých mnoho lidí vlastní i více, konče u těch nejmenších, jako jsou různá čidla, nositelné příslušenství k mobilním telefonům nebo různé mikrokontroléry. Není problém si pomocí Internetu objednat letenku či večeři v luxusní restauraci, vytisknout soubor na tiskárně se síťovou kartou nebo zapnout vytápění domu po cestě z práce.

Takový rozmach Internetu sebou ovšem přináší i velkou míru nebezpečí a zodpovědnosti. Prakticky neustále probíhají útoky na servery či celé počítačové sítě. Typů takovýchto útoků je velmi mnoho a často je velmi obtížné a nákladné se proti nim efektivně bránit. Navíc vymýšlí útočníci stále nové způsoby, jak takováto zabezpečení obejít. Motivy jsou různorodé. Může se jednat o útok s cílem získání citlivých informací, jakožto přístupové údaje do informačního systému, čísla platebních karet atp. nebo snaha o narušení či zablokování správné funkčnosti daného serveru. Velká část útoků pak probíhá pouze jakožto nácvik na jiný útok nebo aby útočník čistě z principu ukázal své schopnosti. Nemusí se však jednat pouze o špatnou činnost. Mnohdy si subjekty (např. banky) platí za tzv. penetrační testy, při kterých se hodnotí jejich počítačové a síťové zabezpečení, tj. jak je subjekt schopen se případným útokům ubránit.

Už dávno se nejedná o útoky ve smyslu opakovaného načítání stránky, dokud se cílový server nezahltí vznikajícími požadavky. Často se jedná o zneužití různých bezpečnostních děr v použitých aplikačních protokolech, podvržení odpovědí od serveru, zachytávání nešifrovaného přenosu atp. Navíc se nejedná o útok cílený z jednoho stroje, ale útočníci mají často vybudovanou celou síť počítačů, které se následně o útok pokoušejí. Takovou síť, nazývanou botnet, je možné si vybudovat například šířením virů, které umožní nad strojem převzít kontrolu.

Z důvodu rostoucího počtu a komplexnosti útoků je nutné zdokonalování ochrany. Mezi nejzákladnější typicky patří firewall a antivirový program. Ty jsou nakonfigurovány buď na koncové stanici nebo na některém z aktivních prvků, např. na routeru (případně na obou). Jejich hlavní pracovní náplní je analýza každého síťového přenosu, který přes tato zařízení prochází a pomoci různých heuristik v něm odhalovat potenciální nebezpečí. Takové řešení má jednu zásadní nevýhodou, kterou jsou vrstvy ISO/OSI modelu, na kterých pracují. Ne-

provádějí analýzu na aplikační vrstvě, na kterou se mnoho útoků soustředí. Taková analýza je výkonnostně velice náročná, jelikož daný software musí provézt jak rozbalení dat aplikační vrstvy od hlaviček nižších vrstev, tak samotné zkoumání škodlivosti těchto dat, což není jednoduchá záležitost. Aplikačních protokolů se na síti vyskytuje velká spousta a navíc je často nutné znát i kontext dané zprávy, tj. předchozí komunikaci.

Pro monitorování síťových toků se dnes nejčastěji využívají systémy založené na architektuře NetFlow a IPFIX[1]. Taková architektura se typicky skládá z několika zařízení. Základem je prvek, který sbírá data o síťové komunikaci. Následuje kolektor, který tato data přijímá a následně zpracuje. Poslední částí je komunikační protokol mezi oběma zařízeními, tvořící jeden z rozdílů mezi zmíněnými architekturami. Největším rozdílem jsou pak informace, se kterými zdroj dat a kolektor pracují. Jak bylo zmíněno na začátku, jsou počítačové sítě využívány stále více zařízeními z celého světa. Je proto nutné umět síťový provoz profilovat, tj. třídit data do jednotlivých profilů. Ty mohou sloužit například pro oddělení síťového provozu na základě aplikačního protokolu, díky čemuž je možné sledovat zátěž a vytvářet statistická data. Kromě toho mohou být data profilována například podle IP prefixů. To je důležité zejména při hledání útočníků při detekovaném útoku.

Cílem této práce je navrhnout systém pro profilování síťových dat pomocí IPFIX mediátoru. Je nutné navrhnout algoritmus pro efektivní filtrování dat pomocí flexibilních filtračních pravidel. Kromě toho je nutné mít možnost specifikovat hierarchii profilů, do kterých se data třídí. Důraz je kladen na co nejefektivnější, ale zároveň lehce konfigurovatelné filtrování. Zároveň je nutné zachovat co největší propustnost samotného mediátoru.

Text práce je rozdělen do 5 částí. Kapitola 2 se zabývá problematikou monitorování sítí. Jsou zde rozebrány různé způsoby pohledu na síť, přičemž hlavní důraz je kladen na monitorování síťových toků. Je zde také popis nástrojů, které se za tímto účelem aktuálně používají, jejich výhody a nedostatky. V kapitole 3 jsou popsány specifikace a požadavky pro profilování síťových dat nad architekturou IPFIX. Popisuje se zde návrh a následná implementace systému pro klasifikaci dat a správu profilů. Dosaženými výsledky se zabývá kapitola 4, kde je zobrazena hlavně výkonnostní a paměťová náročnost implementovaného řešení. Poslední částí je kapitola 5, která obsahuje shrnutí obsahu a hodnocení dosažených výsledků.

[1] Internet Protocol Flow Information Export

Kapitola 2

Monitorování a správa sítí

Jelikož se nároky na počítačové sítě neustále zvyšují a stoupá také počet připojených zařízení, je nutné mít přehled nad fungování sítě. Důležité jsou především informace o tom, co se v síti děje a kdo s kým komunikuje. Monitorování sítě může sloužit například ke zjišťování stavu jednotlivých síťových zařízení nebo propustnosti linek. To lze využít zejména při budování sítě nebo při určování slabých míst za účelem jejich nahrazení výkonnějšími prvky. Dalším důvodem k zavedení technik pro monitorování sítí je centralizace správy. Díky níže zmíněných přístupům ke sledování sítě mohou být potřebné informace shromažďovány na předem určených místech. K těmto datům pak má přístup spravující proces, který je napojen i na sledované prvky, které tak může po analýze dat vzdáleně nastavovat.

Neméně podstatné je zjišťování informací o komunikaci mezi prvky z důvodu bezpečnosti. Proto je analýza dat velmi důležitá. Umožňuje odhalení případného útoku a identifikaci útočníka, případně včasnou reakci správce sítě. K tomu je však potřeba uchovávat nejen informace o stavu jednotlivých síťových prvků, ale také záznamy jejich komunikace.

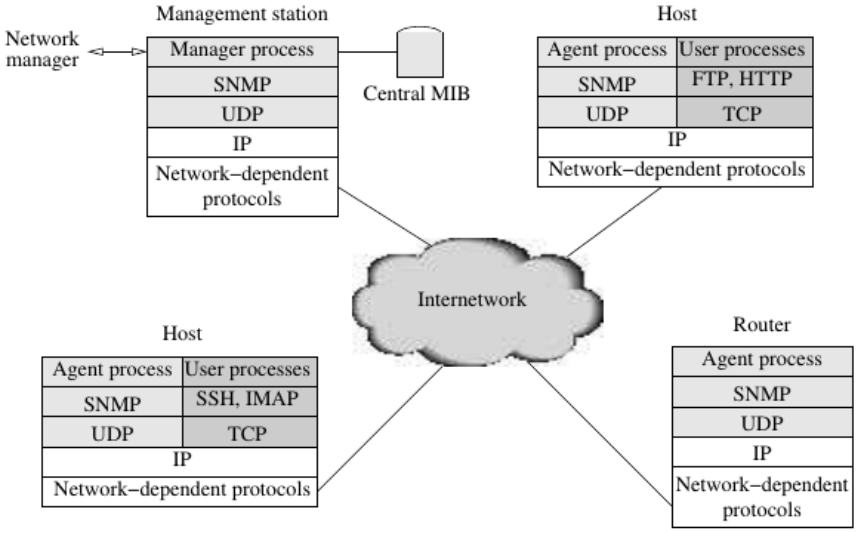
Samotný způsob získávání informací se pak dělí na aktivní a pasivní. U aktivního způsobu dochází k pravidelnému testování dostupnosti zařízení či zjišťování propustnosti linek a zasílání požadavků o informace. Při pasivním přístupu se pak různé statistiky, hlášení o chybách atp. vytvářejí na základě sledované komunikace a navíc mohou sledovaná zařízení sama iniciovat spojení a hlásit své stavové informace nějakému sběrači těchto dat.

2.1 SNMP

Simple Network Management Protocol (SNMP)[2] je jeden ze systémů pro monitorování sítě [9]. Slouží zejména pro získávání informací o jednotlivých stanicích připojených k síti. Ke své činnosti potřebuje několik prvků, jejichž spolupráce je zobrazena na obrázku 2.1:

- agent – software, který shromažďuje informace o jedné konkrétní stanici v sítí a uchovává je v lokální databázi
- manažer – řídící stanice (v jedné sledované síti jich může být i více), jenž žádá agenty o zaslání nashromážděných dat a ta následně ukládá do hlavní databáze
- databáze MIB[1] – hlavní datové úložiště, ve kterém jsou uchovány data od všech agentů
- SNMP protokol – protokol pro přenos dat mezi stanicemi

[1] Management Information Base



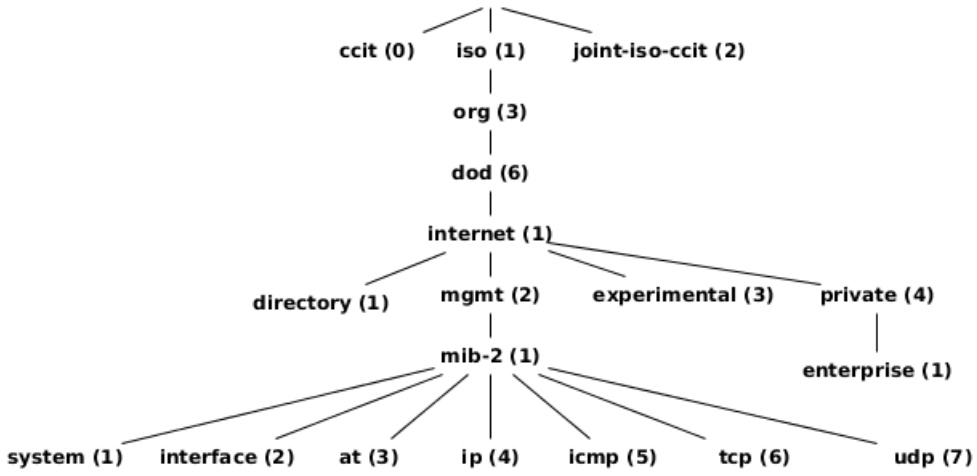
Obrázek 2.1: Architektura systému SNMP

Základním typem komunikace mezi řídící stanicí a agentem je tedy dotaz-odpověď. V některých situacích je však dotazovací mechanismus nevhodný. Jedná se zejména o situace, kdy jeden manažer spravuje mnoho stanic, které navíc obsahují velké množství objektů popisujících jejich stav. Pro zamezení velkého vytížení řídící stanice a navýšení síťového provozu proto existují zprávy typu trap[4]. Jedná se o zprávu generovanou agentem, který následně iniciuje spojení s řídící stanicí a tuto zprávu ji odešle. Trap zpráva typicky obsahuje identifikátor objektu, který ji vygeneroval, adresu agenta, časovou značku, typ zprávy a seznam vlastností a jejich hodnot. K zaslání trap zprávy dochází při předem dohodnutých událostech. Ty mohou být určeny časovou periodou nebo překročením hodnoty nějakého čítače atp. SNMP tak kombinuje aktivní i pasivní způsob monitorování sítě.

Data o jednotlivých stanicích jsou strukturována ve formě objektů. Všechny tyto objekty jsou pak uloženy v hlavní databázi MIB[10]. Objekty jsou jednoznačně identifikovatelné podle OID[2], která jsou uložena ve stromové struktuře, která je znázorněna obrázkem 2.2. Celé jméno daného prvku se tak skládá z identifikátoru předka doplněného o unikátní číslo na aktuální úrovni. Alternativně mohou být názvy vyjádřeny také textově. Například objekt `ip` lze tedy identifikovat jak pomocí řetězce `iso.org.dod.internet.mgmt.mib-2.ip`, tak pomocí čísel ve tvaru `1.3.6.1.2.1.4`.

Velkou výhodou SNMP je, že dokáže získat detailní informace o sledované stanici. Mezi ně patří například název a verze běžícího operačního systému, počet a stav síťových rozhraní, seznam přihlášených uživatelů a mnoho dalších. Oproti jiným technikám pro monitorování provozu však postrádá celkový pohled na sledovanou síť. Dokáže sice určit množství přenesených dat z dané stanice, nelze z nich ale zjistit, kam data putovala. Zároveň také nedokáže určit statistiky z konkrétního časového intervalu (to je nutné dělat ručně, například odečtením uložené hodnoty čítače od nově přijaté), ale pouze za celkovou dobu monitorování.

[2]Object Identifier



Obrázek 2.2: Hierarchie identifikátorů v databázi MIB

2.2 Monitorování toků

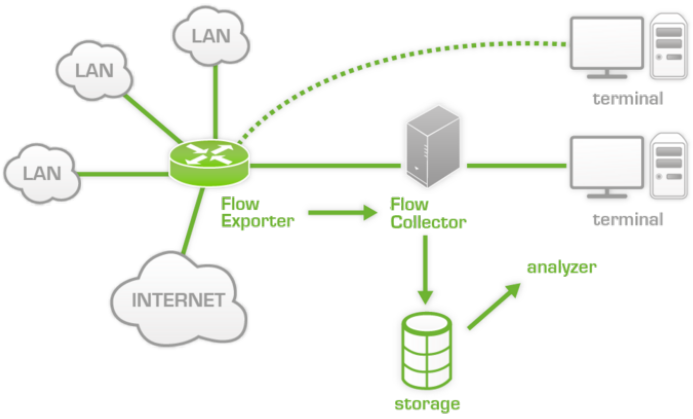
Jiným přístupem k monitorování sítě je zaměření se na toky. Tok[1] (anglicky flow) je definován jako množina paketů, které sdílejí určité vlastnosti. Mezi shodné atributy typicky patří transportní protokol, zdrojová a cílová IP adresa a zdrojový a cílový port. Oproti SNMP poskytuje monitorování toků mnohem detailnější informace o komunikaci mezi síťovými prvky. Díky tokům lze analyzovat složení síťového provozu, což může poskytnout možnosti pro optimalizaci síťové infrastruktury. Zkoumání toků často vede k odhalení různých anomálií v komunikaci, díky čemuž je možné odhalit případné útočníky a oběti a zvýšit tak bezpečnost sítě.

Typická architektura pro monitorování toků je ukázána na obrázku 2.3 a sestává z několika částí:

- exportér – sonda, která zkoumá procházející komunikaci a na jejím základě vytváří nebo aktualizuje toky
- kolektor – zařízení, které sbírá data z exportérů a ukládá je do databáze
- protokol – komunikační protokol pro přenos informací mezi sondou a kolektorem
- analyzátor – systém pro analýzu nasbíraných dat, detekci anomálií atp.

Exportér může být fyzické zařízení nebo software běžící na aktivních prvcích sítě. Postupně analyzuje procházející síťový provoz a vytváří záznamy v tzv. flow cache. Zde jsou uchovány až do skončení toku, po kterém je exportér odešle na kolektor. K ukončení toku může dojít v několika případech:

- ukončení TCP komunikace, tj. při detekci nastaveného příznaku RST nebo FIN
- vypršení aktivního timeoutu, což je maximální možná doba trvání toku
- vypršení neaktivního timeoutu neboli příliš dlouhý interval mezi pakety patřící do jednoho toku



Obrázek 2.3: Architektura pro měření toků

- zaplnění flow cache, kdy je nutné udělat místo pro nový tok

Mezi typické vlastnosti sondy také patří vzorkování a filtrování dat. Vzorkování má význam především při snižování nároků na hardware a může být deterministické nebo náhodné. Pří deterministickém vzorkování je dána perioda, se kterou jsou data zaznamenávána. V tom případě se pak ukládají informace o každém N-tém paketu. Náhodné vzorkování žádnou takovou periodu neuvádí a pakety jsou zpracovány náhodně. To bývá zpravidla přesnější, jelikož je schopné zachytit i pravidelně se objevující data, která mohou být při deterministickém vzorkování ignorována po celou dobu jejich výskytu.

Pro účely filtrace se používají hlavně informace obsažené v hlavičce paketu. Typicky se jedná o ToS (Type of Service). Často se ale provádí i hloubková analýza paketu, kdy je paket klasifikován dle typu přenášených dat.

Data ze sond se zasílají na kolektor, což je software běžící na některé stanici. Dokáže přijímat data od více exportérů a ukládá je do datového úložiště. Zároveň může poskytovat nástroje pro jejich vizualizaci, bezpečnostní analýzu atp.

2.2.1 NetFlow

NetFlow je proprietální systém pro monitorování síťových toků vyvinut společností Cisco Systems. Architektura NetFlow definuje exportér jako aktivní zařízení (router, switch). Toto zařízení vedle své činnosti, ke které je primárně určeno, ještě sbírá informace o tocích a odesílá je kolektoru. Tento přístup je ovšem náročný na výpočetní výkon daného zařízení a tím pádem také na pořizovací cenu. Pro dosažení požadované síťové propustnosti se v dnešní době využívají spíše samostatné sondy, které jsou typicky umístěny na vstupu do sítě. Ty mají za úkol pouze vytváření a exportování informací o tocích.

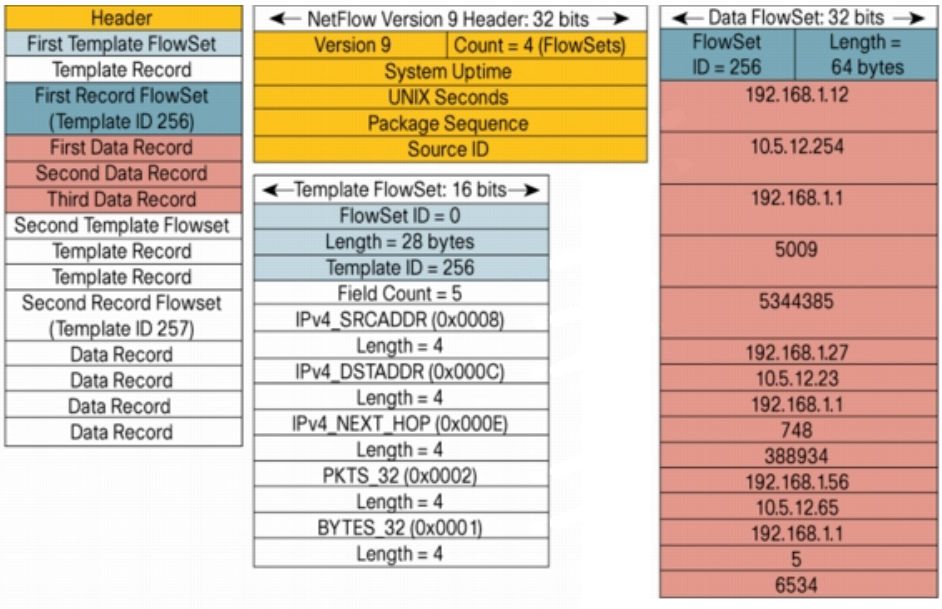
Byla vytvořena celá řada protokolů pro přenos dat mezi sondou a kolektorem, nejvíce se pak využívají především protokoly NetFlow v5 a NetFlow v9[13].

Protokol ve verzi 5 poskytuje základní možnosti pro sběr dat o tocích. Paket obsahuje hlavičku s informacemi o verzi protokolu, počtu obsažených flow záznamů, identifikátor zdroje dat (exportéru) a další údaje, znázorněné v tabulce 2.1. V jednom paketu může být maximálně 30 záznamů, přičemž všechny mají pevně definovanou podobu se základními údaji, mezi které patří například zdrojová a cílová IP adresa a velikost přenesených dat.

Naproti tomu verze 9 přináší podstatně odlišný přístup. Hlavička sice zůstává takřka stejná, ale je zaveden nový typ záznamu – šablona. Ty popisují, jak vypadají jednotlivé

Byty	Obsah	Popis
0-1	version	Číslo identifikující použitý NetFlow protokol
2-3	count	Počet toků v aktuálním paketu (1-30)
4-7	SysUptime	Čas od spuštění exportéru (v milisekundách)
8-11	unix_secs	Čas v sekundách od 1.1.1970
12-15	unix_nsecs	Čas v nanosekundách od 1.1.1970
16-19	flow_sequence	Sekvenční číslo určující celkový dosavadní počet flow záznamů
20	engine_type	Typ exportujícího procesu
21	engine_id	Identifikátor exportéru dat
22-23	sampling_interval	První 2 bity udávají způsob vzorkování dat Zbylých 14 bitů udává hodnotu vzorkovacího intervalu

Tabulka 2.1: Hlavička NetFlow v5 paketu



Obrázek 2.4: Ukázka NetFlow v9 paketu

datové záznamy, tj. jaké informace přenášejí a kolik bytů zabírají. Tím je dosaženo mnohem širších možností využití systému NetFlow, jelikož si sondy samy mohou zvolit, jaká data sbírají a odesílají. Záznamy jsou shlukovány do množin, přičemž množina šablon obsahuje jednotlivé vzory pro datové množiny. Datová množina se pak skládá z několika datových záznamů stejného typu a identifikátoru šablony, která určuje jejich podobu. Jak je možné vidět na obrázku 2.4, typický NetFlow v9 paket tedy obsahuje hlavičku a několik množin se záznamy (ať už datovými či vzorovými).

Jako transportní protokol je nejčastěji použit UDP. To může občas vést ke ztrátě dat, jelikož tok je po odeslání odstraněn z paměti exportéru a při jeho ztrátě během doručování nemůže být obnoven. To je problém zejména u NetFlow v9, kde může docházet k agregaci dat a každý nedoručený paket může velkou mírou ovlivnit analýzu dat. I z toho důvodu je možné využít pro přenos protokol SCTP, který zajišťuje ochranu proti ztrátě dat. Nevýhodou tohoto přístupu je zvýšení výpočetního výkonu obou komunikujících stran.

2.2.2 IPFIX

IPFIX je architektura využívající stejnojmenný protokol, který je standardizovaný. Stejně jako u NetFlow je systém složen z exportéru, kolektoru a protokolu. Hlavním rozdílem je, že už se nejedná o proprietální protokol. Standard byl vytvořen pracovní skupinou IETF. Ten například striktně definuje, jakou funkcionalitu musí splňovat kolektor a která je volitelná. Na rozdíl od NetFlow, které posílá data pomocí UDP či SCTP, umožňuje IPFIX i spojení přes TCP. IPFIX bývá někdy neoficiálně označován jako NetFlow v10.

Historie sahá do roku 2004, kdy vznikl dokument popisující požadavky na protokol IPFIX[11] a začal výběr vhodného kandidáta, ze kterého by měl vycházet. Tím se nakonec stal protokol NetFlow v9, jelikož splňoval mnohé z požadavků (velká míra rozšiřitelnosti, podpora vzorkování atp.). V roce 2008 byl protokol IPFIX standardizován[5] a o rok později vznikl návrh pro celou architekturu[12]. Poslední revizí prošel v roce 2013 a aktuálně jej popisuje RFC 7011[6].

Stejně jako u NetFlow v9 je zpráva složena z hlavičky a libovolného počtu množin záznamů, které se nazývají sady. Může se jednat o množiny šablon nebo dat. Každá sada obsahuje hlavičku, ve které je uveden její typ a také celková velikost v bytech (včetně hlavičky). Každá šablona pak obsahuje identifikátor. Ten musí být větší než než 255. Stejný identifikátor je pak uveden v každé datové sadě, která je touto šablonou definována. Kromě klasických šablon se mohou vyskytovat ještě specifičtější šablony. Označovány jsou jako Options Templates a oproti normálním šablonám, které popisují datové záznamy, poskytují informace o čemkoli jiném. Jejich tvar je velice podobný jako u klasických. Rozdíl je v prvních N položkách (přičemž N je specifikováno v hlavičce šablony) udávající elementy, ke kterým jsou následující informace vztaženy. Typicky se jedná o různé statistické údaje. Příkladem může být celkový počet odeslaných paketů z jedné sondy.

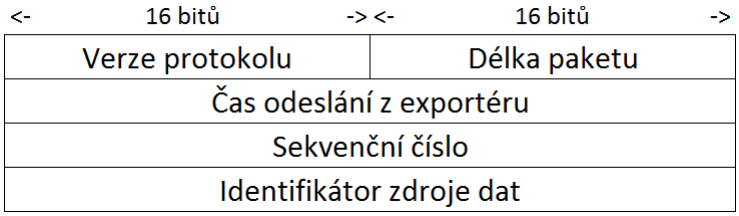
IPFIX řeší také problém položek s proměnnou délkou. Šablony obsahují 2 byty pro určení velikosti elementů. Pevná velikost položky tak může být nejvýše 65535. Tato hodnota je však rezervována pro identifikaci proměnné délky položky. Elementy s proměnnou délkou si nesou informaci o své velikosti v prvním bytu dat. Pokud se velikost do jednoho bytu nevejde, nastaví se na nejvyšší možnou hodnotu (255) a velikost je určena následujícími 2 byty. Díky tomuto mechanismu lze přenášet různá data s proměnnou délkou, jako například URL adresy v HTTP požadavcích.

Dalším odlišením od NetFlow v9 je počet dostupných položek. Oproti NetFlow, které poskytuje 86 různých datových elementů jich organizace IANA pro protokol IPFIX v současné době definuje 433. Navíc je k dispozici ještě Enterprise ID. Jedná se o identifikátor organizace určující význam dané položky. O tento identifikátor si mohou společnosti žádat u organizace IANA. Je to mechanismus, díky kterému si mohou organizace definovat vlastní položky, aniž by docházelo ke kolizím v jejich číslování. Například VUT Brno má registrované číslo 4193[3]. Přítomnost Enterprise čísla identifikuje nastavení nejvyššího bitu v identifikátoru elementu v dané šabloně na hodnotu 1. Při tomto nastavení se za délku políčka ještě uvede 32 bitový Enterprise identifikátor. Takto je dosaženo prakticky neomezeného množství informací, které lze pomocí protokolu IPFIX přenést.

Obrázek 2.5 ukazuje formát hlavičky IPFIX paketu. Jak už bylo zmíněno výše, je odlišná od NetFlow. Následuje popis informací, které obsahuje:

Verze udává číslo použitého protokolu. Odpovídá verzím NetFlow protokolů, u IPFIX je zde nastavena hodnota 10.

[3] http://www.iana.org/assignments/enterprise-numbers/enterprise-numbers



Obrázek 2.5: Hlavička IPFIX paketu

Délka obsahující celkovou velikost paketu v bytech, včetně hlavičky a všech množin za ní následujících.

Čas odeslání nastaven exportujícím procesem, vyjádřen jako 32 bitové čislo v sekundách od 1.1.1970.

Pořadové číslo sloužící k detekci ztrát paketu nebo špatného odeslání z exportéru. Na straně kolektoru probíhá kontrola sekvenčních čísel a to tak, že příští paket by měl mít tuto hodnotu rovnu součtu sekvenčního čísla aktuálního paketu a počtu datových záznamů v něm obsažených.

ID zdroje je 32 bitový identifikátor exportéru, ze kterého byl paket odeslán.

Součásti standardu IPFIX je popis uložení dat do souboru. Data se do něj ukládají stejně, jako když jsou posílána po síti. Toto je mimo jiné možné využít například i při testování kolektoru, kdy je k dispozici neměnný vzorek dat.

Při komunikaci mezi sondou a kolektorem přes UDP je zapotřebí ještě jeden mechanismus pro zajištění správnosti dat. Oba prvky musí mít nastavenou platnost šablon[14]. Ta je vyjádřena buď v počtu sekund nebo v počtu paketů. Po uplynutí dané platnosti (tj. po vypršení časového intervalu nebo přenesení daného počtu paketů) musí být z exportéru šablona znovu odeslána, jinak ji bude kolektor považovat za neplatnou. Pokud ji tedy exportér neobnoví a kolektor přijme data, která se na ni odkazují, nebude schopen je správně zpracovat. Je velmi důležité tyto hodnoty vybrat co nejpečlivěji, aby nedocházelo ke zbytečnému navyšování síťové zátěže častým obnovováním šablon a zároveň aby nebyly velké ztráty dat z důvodu neúspěšného přenesení obnovených šablon mezi exportérem a kolektorem. Toto opatření je nutné z toho důvodu, že přenos přes UDP negarantuje spolehlivý přenos datagramů, takže zprávy mohou přijít na cílovou stanici v jiném pořadí, než byly odeslané ze zdroje, nebo nemusí přijít vůbec.

2.3 NetFlow kolektor

Jedním z NetFlow kolektorů je nfdump[4]. Jedná se o sadu nástrojů pro realizaci sběru, profilování a analýzy NetFlow dat, podporující protokoly verze 5, 7 a 9. nfdump je ovládán z příkazové řádky a k ukládání dat využívá soubory s daty rozdělenými dle časového intervalu (např. po 5 minutách). Kromě proudového zpracování je nfdump zaměřen i na práci s daty již uloženými na disku. Umožňuje vytvářet různé dotazy a pohledy nad dříve získanými informacemi. K vizualizaci zpracovaných dat je využit webový nástroj NfSen[5].

[4] http://nfdump.sourceforge.net/
[5] http://sourceforge.net/projects/nfsen/

Množství přenesených dat po internetu neustále narůstá, stejně jako počet komunikujících zařízení. Tím se také zvyšuje počet NetFlow záznamů, které typicky kolektor musí být schopen zpracovat. Klíčovou funkci při detekci bezpečnostních hrozeb či monitorování počtu přenesených dat hraje informace o jejich typu a zdroji. K tomuto účelu slouží profilování dat, které se stará o klasifikaci toků do jednotlivých profilů, díky kterým je možné rozdělit data do menších celků. Zároveň je díky profilování dat možné sledovat vytížení sítě, nejčastěji používané protokoly atp.

Data jsou do profilů nejčastěji roztříděna dle nějakého filtračního pravidla. To se aplikuje na hlavičku IP datagramu nebo přímo na data, která jsou jím přenášena. V rámci IP hlavičky se pak jedná hlavně o adresu zdroje dat, z hlediska toků se nejčastěji filtruje na základě zdrojové a cílové IP adresy. Nástroj nfdump pro správu profilů využívá webové rozhraní aplikace NfSen. Data jsou pak při ukládání na disk uchovávána právě dle profilů.

NfSen pro filtrování dat používá profily, které dále mohou obsahovat kanály. Profil je zde brán jako zapouzdření určité sady kanálů. Každý kanál pak obsahuje seznam zdrojů a filtrační pravidlo. Zdroje určují, odkud jsou do kanálu posílána data. Filtrační pravidlo určuje, zda-li data spadají do daného kanálu. Příslušnost dat k profilu je pak určena příslušností alespoň k jednomu jeho kanálu. Příkladem filtračního pravidla pro určení, zda-li spadá záznam do kanálu, může být `tcp and (src ip 172.16.17.18 or dst ip 172.16.17.19)`. Takovému pravidlu odpovídají všechna data přenesena přes TCP ze zdrojové IPv4 adresy 172.16.17.18 na cílovou adresu 172.16.17.19. Mimo to mohou být profily hierarchicky uspořádány a kromě kanálu mohou tedy obsahovat i další podprofily. Zdroje kanálů pak určují názvy kanálů rodičovského profilu.

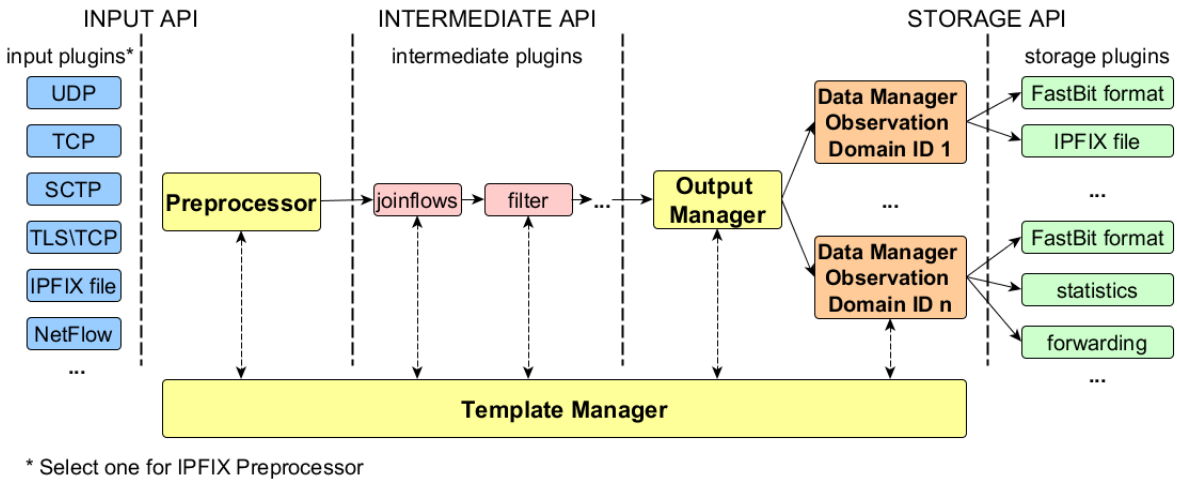
Vedle klasických profilů existují ještě 2 speciální typy - živý (live) a stínový (shadow) profil. Live profil existuje vždy a ukládají se do něj všechna data, která kolektorem (respektive nástrojem pro vytváření a správu profilů) prošla. Kanály tohoto profilu nenesou informaci o zdroji, jelikož přijímají vše. Stínový profil slouží zejména k uložení sekundárních dat, tj. různých statistických údajů o velikosti a počtu přenesených dat atp. Shadow profil již neobsahuje samotné NetFlow záznamy. Toto uživateli umožňuje například počítání různých statistických údajů, aniž by musela být uložena samotná data. V těchto případech je tak dosaženo velké úspory diskového prostoru.

Kromě vytváření a správy profilů umožňuje NfSen i vizualizaci zpracovaných dat. Při vytváření jednotlivých profilů a kanálů je možné specifikovat např. barvu, kterou jsou poté jejich data vykreslována.

2.4 IPFIX kolektor a mediátor

IPFIX kolektor je nástroj, který je schopen přijímat data v IPFIX formátu od jednoho a více exportérů[6]. Jeho primární činností je uložení dat v požadovaném formátu. Musí být schopen přijímat požadavky na spojení od jednotlivých exportérů. Taktéž by měl poskytovat podporu pro kontrolování sekvenčních čísel jednotlivých paketů a hlásit či zaznamenávat případné nesrovnalosti. Zároveň musí být schopen detekovat poškozené zprávy. Takové pakety mohou vzniknout např. v případě, kdy je velikost množiny větší než velikost celkové zprávy nebo je paket menší než velikost IPFIX hlavičky. Při detekci takové zprávy by měl tuto událost opět hlásit či zaznamenat a zprávu ignorovat.

Vedle kolektoru pak ještě existuje tzv. mediátor[8]. Jeho funkce spočívá v příjmu dat, jejich zpracování pomocí mediačních procesů a následném odeslání na další zařízení, kterým může být kolektor nebo další mediátor. Některé kolektory mají implementovanou podporu pro mediátor již v sobě. Mezi takové patří například open source nástroj IPFIXcol[3] vyvíjen



Obrázek 2.6: Architektura nástroje IPFIXcol

organizací Cesnet z. s. p. o. Ten je charakterizován vysokou modularitou a je složen z jádra kolektoru a vstupních, mediačních a výstupních pluginů (viz. obrázek 2.6).

Jádro kolektoru slouží ke zpracování IPFIX zpráv a stará se o správu šablon. Zajišťuje hlídání životnosti šablon (při přenosu přes UDP), detekci datových záznamů s odkazem na neznámou šablonu a kontrolu sekvenčních čísel.

Vstupní pluginy zajišťují příjem dat pro kolektor. Typicky se jedná o moduly, které naslouchají nově příchozím požadavkům od exportérů nebo čtou data z disku. Kolektor musí být schopen přijímat data přes UDP, TCP i SCTP. IPFIXcol pak umožňuje mimo samotné IPFIX zprávy i zpracování protokolu NetFlow ve verzi 5 a 9. V tom případě je nutná konverze těchto zpráv do správného tvaru, jelikož jádro kolektoru již zpracovávaná data nijak nerozlišuje a od všech předpokládá, že jsou typu IPFIX.

Převod zpráv z NetFlow formátu není nijak složitý. U verze 5 je zapotřebí vytvořit šablonu, která tento typ popisuje. Zároveň je nutné při přenosu přes UDP dodržet pravidelné odesílání této šablony tak, jak by to prováděl exportér u IPFIX dat. U verze 9 se jedná jen o úpravu některých elementů (např. časové značky mají v NetFlow formátu 32 bitů, kdežto IPFIX očekává 64 bitové).

Výstupní moduly slouží k závěrečnému zpracování dat. Typicky se jedná o jejich uložení na fyzické úložiště. K tomu mohou sloužit různé databáze (například PostgreSQL[6]) nebo mohou být data uchována přímo v IPFIX formátu tak, jak to popisuje RFC 5655[15]. Kromě uložení mohou ale být data například odeslána po síti, a to v IPFIX či jiném formátu. Právě odeslání dat k jinému zařízení je jednou z hlavních částí definice mediátoru a odlišuje se tím od kolektoru.

Poslední částí jsou mediační pluginy. Ty mají na vstupu IPFIX data zpracovaná jádrem kolektoru, nad kterými mohou vykonat nějakou činnost a opět je odeslat dále (dalšímu mediačnímu procesu či výstupní části kolektoru). Jejich práce může být prakticky jakákoli a nemusí se nutně jednat o modifikaci dat. Mohou nad daty například počítat různé statistiky, které jsou uchovávány v databázi pro pozdější analýzu. Modifikace, které mohou na data aplikovat jsou sice neomezené, na jejich výstupu ovšem musí být opět validní IPFIX data (nebo nic). Nesmí se stát, že by do další části zpracování odeslaly poškozená data.

[6] http://www.postgresql.org/

Mezi možné transformace dat patří například spojování toků z více sond tak, aby se výstupním modulům tato data jevila jako od jednoho zdroje. To může probíhat například na základě ODID (Observation Domain ID). Takový modul má opodstatnění například v tom, že se díky spojování toků sníží množství uložených dat, jelikož se často stává, že šablony z různých exportérů jsou totožné. I o to se tedy spojovací modul stará a je tím zajištěno sjednocení takových duplicitních šablon.

Z popsaných vlastností je zřejmé, že IPFIX mediátor je díky modularitě a podpoře více protokolů silným nástrojem pro sběr a transformaci síťových dat. Jeho hlavní předností oproti NetFlow kolektoru je právě variabilita IPFIX protokolu. Ten je použitelný pro přenos jakýchkoli dat. V čem ovšem IPFIX mediátor zaostává, je analýza dat. Pro efektivní analýzu je ale užitečné mít data nejprve rozdělena do profilů. Mediátor však neposkytuje žádný mechanismus, jak tohoto dosáhnout. Na druhou stranu nfdump, respektive NfSen, obsahuje propracovanou podporu pro filtraci a profilování dat, nicméně jeho využitelnost je omezena protokoly NetFlow. Ty jsou oproti IPFIX protokolu méně flexibilní a rozšiřitelné. Hlavně však neumožňují ukládání dat z aplikační vrstvy.

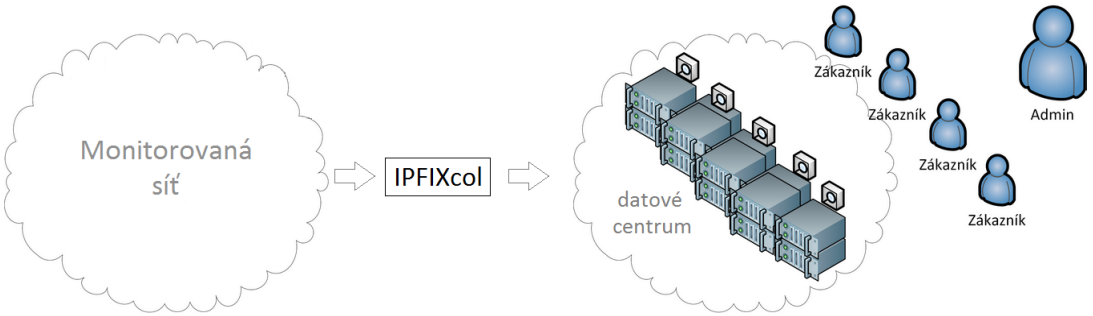
Kapitola 3

Profilování v IPFIX mediátoru

Jak již bylo zmíněno u NetFlow kolektoru, je profilování dat velmi důležitým prvek, zejména prochází-li kolektorem velké množství dat z různých zdrojů. Umožňuje například rozdělení síťového provozu dle různých aplikačních protokolů. Lze tak zjistit, kolik procent zátěže tvoří DNS komunikace, HTTP požadavky, telekonference atd. Bohužel, IPFIXcol v současné době nemá pro tuto funkcionalitu podporu. Proto je nutné navrhnout a implementovat vhodné řešení, které by profilování zajišťovalo.

3.1 Specifikace požadavků

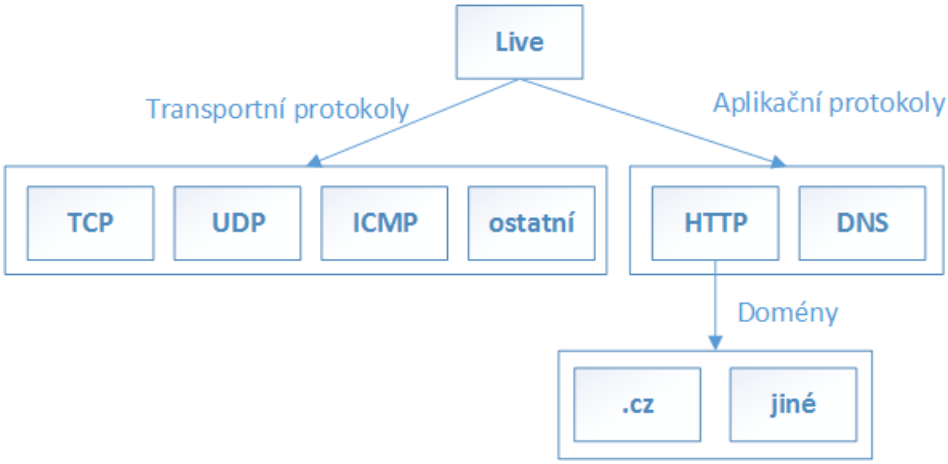
Nejprve je nutné stanovit si požadavky na systém profilování. K tomu je vhodné určit typické příklady použití. První případ použití je účtování za přenesená data či kontrola dodržování podmínek uvedených ve smlouvě mezi poskytovatelem služby a jejím konzumentem[1]. Dalším typickým případem použití je, kdy provozovatel služeb má zájem poskytnout náhled na síťová data svým zákazníkům. Každému zákazníkovi umožní poskytovatel pohled pouze na svá data. Zároveň správce poskytovatele vidí data od všech zákazníků. Takový případ použití je znázorněn na obrázku 3.1. Zákazník musí být identifikovatelný pomocí několika údajů, kterými jsou zdroj dat, tj. Observation Domain ID a IP adresa exportéru, ze kterého data přišla, dále pak IP prefix nacházející se přímo v datech, tj. zdrojová či cílová adresa v konkrétním datovém záznamu a nakonec MPLS[2] label. Je nutné mít k dispozici více než 1 úroveň profilů, přičemž každý zákazník si může definovat své vlastní hierarchie profilů. Samotný počet zákazníků a profilů nesmí být systémem profilování nijak omezen.



Obrázek 3.1: Příklad použití profilování v IPFIX mediátoru

[1] SLA - Service-level agreement
[2] Multiprotocol Label Switching

Další případ použití je již výše zmíněné profilování dle aplikačních protokolů či jiných ukazatelů, díky čemuž je možné získat podrobnější statistiky ze zpracovaných dat. Výsledkem jsou detailnější informace o rozložení zátěže, které mohou sloužit k rekonfiguraci síťových prvků za účelem vyšší propustnosti nejčastěji se vyskytujícího provozu. Typicky je důležité rozlišit toky dle typu provozu. Live profil tedy bude obsahovat několik podprofilů, kde každý reprezentuje určitý transportní protokol. Nejčastěji se pro statistiky provoz dělí na TCP, UDP, ICMP a zbytek. Vedle toho je důležité mít filtraci i dle aplikačních protokolů. V tomto případě jsou vhodnými kandidáty protokoly HTTP a DNS, tj. porty 80 a 53. DNS toky jsou důležité zejména pro odhalování potenciálních útočníků. Podezřelé může být například výrazně zvýšený provoz. Takové podezření může administrátora sítě či nějaký detekční modul vézt k detailnějšímu zkoumání dění, což může mít za následek například objevení pokusů o podvržení DNS odpovědi atp. HTTP provoz je zajímavý zejména z pohledu statistik, jako například nejčastěji navštěvované domény, nejpoužívanější metody a jiné statistické ukazatele. Ukázka takové hierarchie profilů je ukázána na obrázku 3.2.



Obrázek 3.2: Příklad profilování dle typu provozu

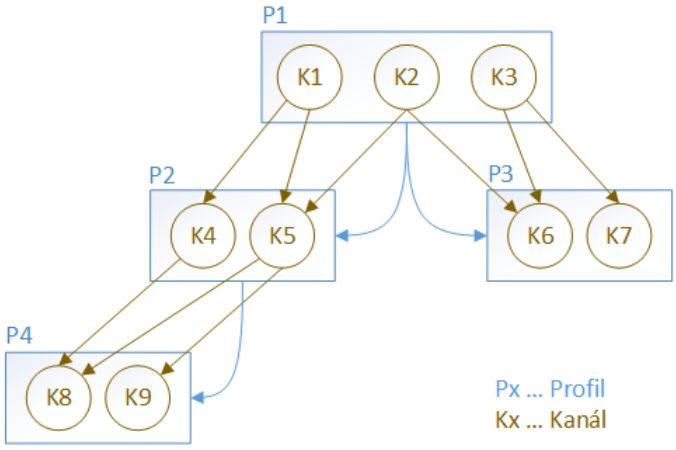
Je nutné vymyslet podobu samotných profilů. Pomineme-li pojmy jako organizace a zákazník, uvedené v příkladech použití, získáme 2 základní požadavky:

- profily musí být možné hierarchicky zanořovat, tzn. vytvářet podprofily na libovolné úrovni
- každý profil může obsahovat filtrační pravidlo, kterým se určuje příslušnost záznamu k danému profilu

Dalším omezením plynoucím z případů užití je zdroj dat každého profilu. Budeme-li předpokládat, že na nejvyšší úrovni jsou data rozdělená do různých organizací, musí mít na každé další úrovni profil přístup pouze k datům dané organizace. Obecně se tedy jedná o to, že každý profil má přístup k datům pouze z jeho rodičovského profilu.

3.2 Návrh a implementace

Prvním problémem při navrhování systému pro profilování dat je jeho umístění v rámci kolektoru. Je zde několik možnosti, jak může být řešení implementováno. Může se jednat



Obrázek 3.3: Příklad hierarchie profilů a kanálů

o vstupní plugin, což ale není nejlepší způsob, jelikož hlavní činností vstupních modulů je transformace dat na jejich vnitřní reprezentaci jakožto IPFIX paket. Vstupem profilovacího modulu by měla být již předzpracovaná data. Další možností je výstupní plugin. Jak ale vyplývá ze specifikace požadavků, mohou být vyprofilovaná data dále ukládána do různých datových úložišť. Za tímto účelem by musel profilovací plugin na výstupní pozici obsahovat i samotnou správu ukládání dat, čímž by se ale jeho činnost limitovala na uložení pouze v implementovaném formátu. Vhodnou možností je tedy modul realizovat jakožto mediační plugin. Ty jsou obecně použity za účelem modifikace IPFIX zpráv, počítání různých statistik atp., což plně vyhovuje potřebám profilovacího modulu. Zde ovšem přichází na řadu další problém. Je vhodné informace o profilech sdílet mezi více moduly. Pokud by například při profilování existoval výstupní plugin, který bude data ukládat dle jednotlivých profilů, musí mít přístup ke stejným datům, jako profilovací modul. Navržené řešení se tedy skládá ze dvou částí. První z nich je správa profilů. Ta je zařazena přímo do jádra kolektoru, který jednotlivým modulům poskytuje rozhraní pro práci s profily. Takto je celá konfigurace načtena na jediném místě a všem modulům jsou dostupná shodná a konzistentní data. Druhou částí je samotné profilování dat. O něj se stará mediační plugin, který využívá rozhraní pro správu profilů. Zde se k datovým záznamům IPFIX paketu přiřazují informace o odpovídajících profilech.

Po shrnutí všech požadavků a omezení, přichází v úvahu podobný systém, jako je použit v nástroji NfSen. Každý profil bude obsahovat strukturované informace, mezi které patří identifikátor profilu a seznam kanálů náležících do daného profilu. Všechna příchozí data na kolektor náleží výchozímu profilu live. Profil live není možné zrušit a vždy musí být definován a existovat v mediátoru. Existující profil může kromě jiného obsahovat i další podprofily.

Kanál kromě jména určuje i samotné filtrovací pravidlo, které je aplikováno na datový záznam. Navíc je možné u kanálu specifikovat zdroje, ze kterých data čerpá. Zdrojem je zde myšlen textový identifikátor kanálu z rodičovského profilu. Ve výsledku se tak jedná primárně o stromovou hierarchii kanálů, přičemž jejich určité skupiny jsou zabaleny do profilů, které určují rodičovské vazby. Příklad takové hierarchie je ukázán na obrázku 3.3, kde šipky znázorňují zdroje, které do daného kanálu dodávají data. Kanály na nejvyšší úrovní zdroje uvedeny nemají, jelikož se jedná o kanály live profilu, který přijímá všechna data.

Postup zpracování každého paketu je tedy následující:

1. Každý záznam v IPFIX paketu je předán live profilu.
2. Live profil postupně předá záznam všem jeho kanálům.
3. Každý kanál na záznam aplikuje filtrační pravidlo. Pokud je vyhodnoceno kladně, je kanál přidán do metadat a následně předá záznam všem kanálům, pro které je zdrojem dat.

3.2.1 Správa profilů

Správa profilů je implementována v samotném mediátoru. Vstupem systému pro zpracování profilů je konfigurační soubor ve formátu XML (Extensible Markup Language). Ten obsahuje veškeré informace o profilech a jejich kanálech. Profily jsou hierarchicky zanořovány a vznikají tak rodičovské vztahy. Každý profil obsahuje jméno, které musí být mezi sourozenci (tj. profily se shodným rodičovským profilem) unikátní. Dále se v něm vyskytují jednotlivé kanály a případně podprofily. Kanál, stejně jako profil, obsahuje jméno. To musí být unikátní v rámci profilu, do kterého kanál patří. Kromě jména je v kanálu seznam zdrojů a filtrační pravidlo.

Profily jsou vytvářeny jako stromová struktura. Při zpracování konfigurace je vytváření profilů rekurzivně zanořováno. Každý profil obsahuje identifikátor, seznam potomků a seznam kanálů. Třída reprezentující kanál obsahuje kromě jména a filtračního pravidla ještě seznam zdrojů a odběratelů. Do seznamu odběratelů se mohou registrovat všechny kanály z přímého potomka profilu, ve kterém se daný kanál nachází. To slouží k efektivnějšímu profilování dat, jak je popsáno v následující kapitole.

Důležitým prvkem je způsob identifikace profilů a kanálů z jednotlivých modulů. Jednou možností je odkazování se na ně pomocí jména. To je však unikátní pouze v rámci rodičovského profilu, proto je tento způsob nepoužitelný. Další možností je úplný název neboli konkatenace jména profilů se jmény všech jeho předků. Tím se získá unikátní identifikátor, nicméně zpracování takového řetězce a následná lokalizace profilu či kanálu by byla časově neefektivní operací. Navíc ani tento identifikátor nemusí být unikátní. IPFIXcol totiž podporuje rekonfiguraci za běhu procesu. Pokud obdrží příslušný signál, dojde k opětovnému načtení těch modulů, kterým se změnila konfigurace, případně byly přidány či odebrány. Stejně tak se kontroluje změna cesty k souboru s nastavením profilů, případně se porovnává čas poslední modifikace s jeho předchozím načtením. Při rekonfiguraci tak může být načten úplně jiný strom profilů. Poté může nastat situace, kdy s profily pracuje více modulů. Pokud by je adresovaly jménem, mohly by pracovat s úplně jinými či neexistujícími profily než předešlé moduly, jelikož během předání dat došlo k rekonfiguraci mediátoru a aktualizaci stromu profilů. Nejjednodušším řešením je tedy pracovat přímo s ukazateli na profily a kanály. Mediátor si zároveň uchovává informace o několika předchozích konfiguracích profilů. To z toho důvodu, že data, která byla profilovacím modulem zpracovávána před rekonfigurací mediátoru, ještě mohou být zpracovávána jiným pluginem a ten se tedy může odkazovat na staré profily.

Pro jednodušší implementaci je tento systém napsán v jazyce C++ jako statická knihovna. Navíc obsahuje rozhraní, které poskytuje mapování této funkcionality do jazyka C, v němž je naprogramován mediátor. Ukazatele na jednotlivé profily a kanály jsou do prostředí mediátoru a modulů přetypovány na typ `void*` a nehrozí tak jejich modifikace mimo knihovnu.

3.2.2 Profilování dat

IPFIX mediátor ke každému paketu, který jím prochází, přidává ještě sekundární data, neboli metadata. Ta jsou vztažena ke každému datovému záznamu zvlášť a obsahují dodatečné informace. Může se jednat například o geolokační údaje. Úkolem profilovacího modulu je k těmto metadatům přidat informace o odpovídajících profilech a kanálech. Pro větší efektivitu a menší paměťovou a výpočetní náročnost jsou do seznamu zahrnuty pouze kanály. Z každého kanálu je totiž možné získat informaci o profilu, do kterého patří. Přidávání profilů do metadat by tak bylo zbytečné.

Preprocesor mediátoru do IPFIX zprávy přidává odkaz na aktuální kořenový profil. Po přijetí zprávy profilovacím modulem je testována náležitost každého datového záznamu do tohoto profilu a všech jeho potomků. Příslušnost k profilu znamená, že datový záznam spadá do alespoň jednoho jeho kanálu. Každý kanál na záznam aplikuje příslušné filtrovací pravidlo. Pokud danému pravidlu vyhovuje, vloží se odkaz na kanál do seznamu kanálu příslušného záznamu. Následně se projde seznam všech odběratelů kanálu a každému je předán datový záznam. Takto se rekurzivně projde hierarchie všech odběratelů a výsledkem je seznam kanálů, do kterých záznam spadá. Profily v tuto chvíli představují pouze zapouzdření skupin kanálů. Výjimku tvoří kořenový profil, jenž předává záznam všem svým kanálům. Jedná se tedy o procházení stromu kanálů do hloubky. Při vytváření seznamu vyhovujících kanálů je kladen důraz na to, aby do něj nebyl jeden kanál uložen vícenásobně. K tomu by mohlo dojít, pokud datový záznam vyhovuje hned několika zdrojům daného kanálu. V tom případě je záznam kanálem sice zpracován několikrát, ale do metadat se vloží pouze jednou. Tímto je potlačena redundance dat, což má za následek nižší paměťové nároky.

Obrázek 3.4: Seznam kanálů v metadatech

Seznam kanálů, kterým vyhovuje datový záznam, je transformován na pole ukazatelů. Konec tohoto pole je indikován hodnotou `NULL`. Do metadat příslušného záznamu je vložen odkaz na první prvek tohoto pole, jak je znázorněno obrázkem 3.4.

3.2.3 Filtrování dat

Nejpodstatnější částí profilování dat je tvorba filtračních pravidel a jejich aplikace na datové záznamy. Pro zpracování pravidel v konfiguraci profilů jsou využity nástroje flex[3] a bison[4]. Tyto nástroje umožňují automatické generování lexikálního a syntaktického analyzátoru.

[3] http://flex.sourceforge.net/
[4] http://www.gnu.org/software/bison/

Díky tomu je možné za pomoci relativně jednoduchých instrukcí vytvořit účinný prostředek pro zadávání a zpracování filtračních pravidel.

Filtrační pravidla sestávají z několika částí:

Hlavičkové položky, neboli údaje společné pro celý IPFIX paket. Může se jednat o informace z hlavičky IPFIX paketu či celého IP datagramu. Podporované položky jsou Observation Domain ID, zdrojová a cílová IP adresa zdroje dat a zdrojový a cílový port.

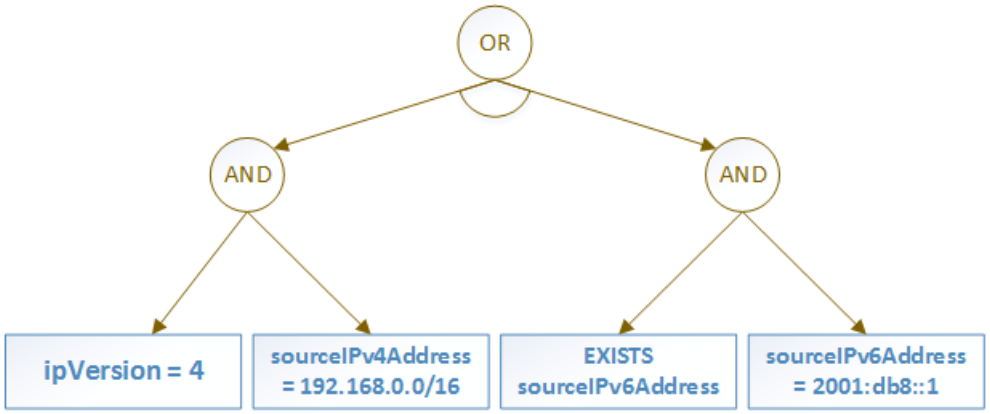
Datové položky, což jsou elementy obsažené v jednotlivých IPFIX záznamech. Jsou dvě možnosti, jak se lze na tato data odkazovat. První z nich je zadání názvu definovaného v konfiguračním souboru popisujícím jednotlivé elementy. Tento soubor je načítán samotným mediátorem a je tak společný pro všechny moduly. Obsahuje popis mnoha datových položek a lze jej rozšiřovat o další. Každý element je popsán pomocí kombinace Enterprise čísla a svého ID. Dále je mu přiřazen název, datový typ a sémantika sloužící pro logické rozčlenění elementů. Druhou možností, jak se na datovou položku odkazovat, je zadání přímo jeho identifikátoru ve tvaru `e<Enterprise>id<ID>`. Seznam elementů pro filtrování tak není nijak omezen.

Hodnoty položek různého typu. Popis jednotlivých typů bude popsán dále.

Relační operátory pro porovnání datových či hlavičkových položek se zadanou hodnotou. Podporovány jsou `<`, `>`, `<=`, `>=`, `=`, `==` a `!=`.

Klíčová slova, mezi která se řadí např. logické operátory pro konkatenaci jednotlivých výrazů. Tyto operátory mohou být zapsány jak slovně, např. `AND` a `OR`, tak i ve stylu programovacích jazyků, tedy `&&` a `||`. Výjimku tvoří negace podporující pouze slovní zápis `NOT`. Dalším klíčovým slovem je `EXISTS`, sloužící pro testování přítomnosti dané položky v datovém záznamu.

Typický výraz filtračního pravidla má tvar `<položka> <relační operátor> <hodnota>`, případně `EXISTS <položka>`. Jednotlivé výrazy jsou pak spojovány pomocí logických operátorů. Navíc mohou být uzavírány do kulatých závorek, díky čemuž lze vytvořit prakticky jakýkoli výraz. Filtrační pravidlo je postupně transformováno do binární stromové struktury. Jedná se v podstatě o AND/OR strom. To znamená, že jednotlivé uzly se rozlišují dle typu. Jedná-li se o AND uzel, pak je jeho platnost splněna, jsou-li platné všechny jeho podstromy. OR uzel vyžaduje platnost alespoň jednoho podstromu. Posledním typem jsou pak hodnotové uzly. Jsou to všechny listy stromu a obsahují již samotné porovnání položky s hodnotou či test na její přítomnost. Na obrázku 3.5 lze vidět tvar filtračního stromu pro pravidlo `(ipVersion = 4 AND sourceIPv4 address = 192.168.0.0/16) || (EXISTS sourceIPv6Address and sourceIPv6Address = 2001:db8::1)`.



Obrázek 3.5: Příklad filtračního stromu

Velice důležitou součástí jsou hodnoty, se kterými lze položky porovnávat. V implementovaném řešení je velký důraz kladen na podporu co nejpřirozenějších formátů. To znamená, že například IP adresy lze zadávat v jejich textovém tvaru. Zde je výčet podporovaných typů hodnot:

Desítkové číslo s příponou, neboli hodnota celočíselného typu následována nepovinnou příponou udávající řád. Mezi přípony se řadí `K`, `M`, `G` a `T`.

Hexadecimální číslo ve formátu 0x… pro jednodušší zadávání hodnot v šestnáctkové soustavě.

IP adresa verze 4 i 6 v textovém formátu. U IP adresy verze 6 je podporován i její zkrácený zápis.

IP adresa s maskou sítě, což je užitečné zejména při zadávání IP prefixů. Ukázku tohoto zápisu pro IPv4 lze vidět na obrázku 3.5.

Časová značka, díky které lze jednoduše filtrovat například dle času začátku toku.

Regulární výraz ohraničen mezi dvojicí znaků `"/` a `/"` poskytující možnost filtrování dle výrazů stejných jako v systémech UNIX.

Textový řetězec, umožňující např. filtrování adres v HTTP požadavcích.

Při filtrování je strom rekurzivně procházen a u každého uzlu se vyhodnocuje jeho podmínka vzhledem k danému datovému záznamu.

Kapitola 4

Výsledky

Nejprve bylo nutné zjistit, zda implementované řešení funguje správně. Mediátor byl nastaven na profilování dle transportních a aplikačních protokolů. HTTP provoz byl dále profilován dle domén v požadavcích. Jedná se o podobnou hierarchii jako v jednom případu použití v kapitole 3.2, jen obohacenou o další profily. Jako zdroj sloužil několik minut dlouhý vzorek dat zachycen na testovací sondě sbírající data ze sítě CESNET. Podrobnější informace o datech jsou v tabulce 4.1.

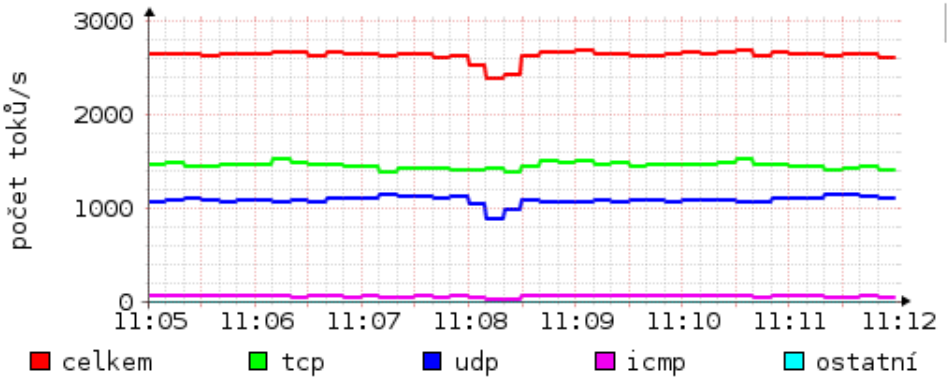
Umístění mediátoru	sonda v síti CESNET
Čas zachycení prvního paketu	11:04:58 2.4.2015
Čas zachycení posledního paketu	11:12:04 2.4.2015
Doba trvání	7 minut 6 sekund
Počet paketů	17 687 936
Počet bytů	13 584 566 818
Počet toků	1 128 900
Průměrná velikost paketu	2 998 B

Tabulka 4.1: Testovací sada dat

Statistiky jednotlivých profilů a kanálů jsou ukládány do RRD[1] databáze v intervalu 10 sekund. Stejný vzorek dat byl testován také v nástroji NfSen. Výsledná data obou nástrojů byla totožná. Zároveň byly na data aplikovány skripty, které vyloučily přítomnost nevyhovujících záznamů. Na základě provedených testů lze tedy předpokládat, že profilování funguje správně.

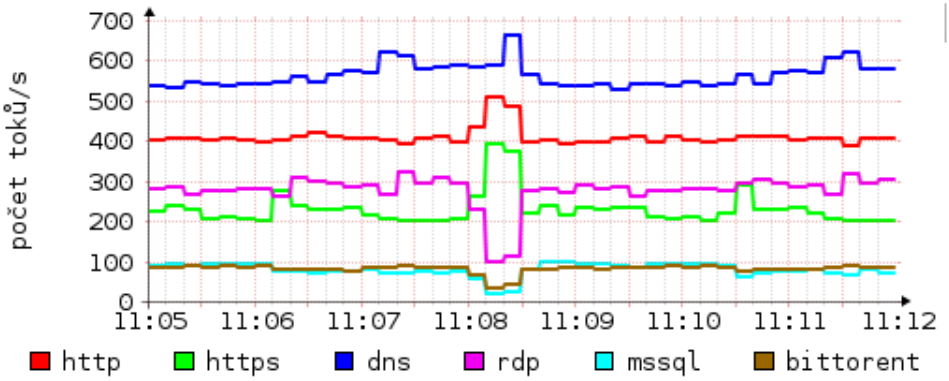
Obrázek 4.1 ukazuje graf rozdělení toků dle transportních protokolů. Vodorovná osa znázorňuje čas, svislá pak počet toků daného protokolu, které mediátorem prošly ze 1 sekundu. TCP i UDP mají v testovaném síťovém provozu velice podobný podíl a společně tvoří většinu celkového provozu.

[1] http://oss.oetiker.ch/rrdtool/



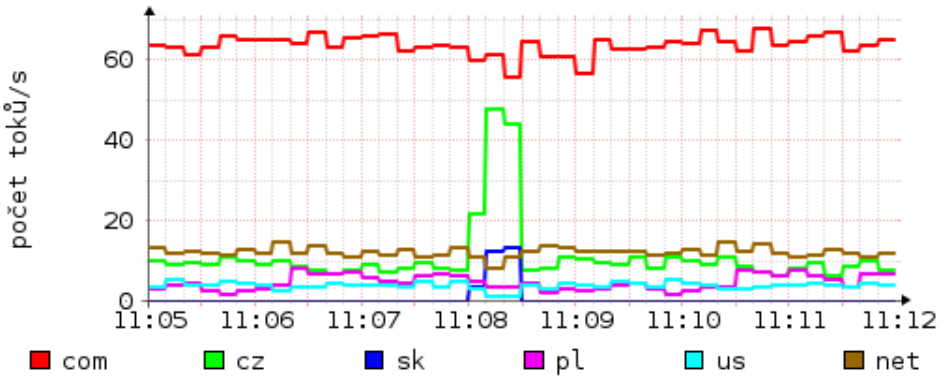
Obrázek 4.1: Typ provozu dle transportních protokolů

Jak je vidět v obrázku 4.2, profily aplikačních protokolů HTTP a DNS zabírají v celkovém provozu největší část. Dále jsou zde zobrazeny další nejčastěji se vyskytující se služby. Zajímavý podíl má např. služba bittorent běžící na portu 6881.



Obrázek 4.2: Rozdělení provozu dle aplikačních protokolů

Posledním pohledem na data jsou informace o doménách v HTTP požadavcích. Do grafu na obrázku 4.3 byly vyneseny informace o několika nejpoužívanějších z nich. Je patrné, že doména `com` tvoří většinu celkových HTTP požadavků. Velký nárůst požadavků s doménami `cz` a `sk` v čase 11:08 lze pozorovat i v předchozím grafu, kde se v témže čase zvýšil podíl HTTP provozu.



Obrázek 4.3: Domény v HTTP požadavcích

Další otázkou je efektivita profilování. Je nutné, aby byl chod kolektoru profilováním co nejméně omezen. Pro tento účel jsem vytvořil několik situací, u kterých byla testována výkonnost v podobě maximálního počtu zpracovaných toků za vteřinu. Dále jsem otestoval paměťovou náročnost. Konkrétně počet alokací a celkovou velikost alokované paměti při zpracování více než 3 milionů toků.

Pro testování propustnosti a paměťové náročnosti jsem aplikoval následující scénáře:

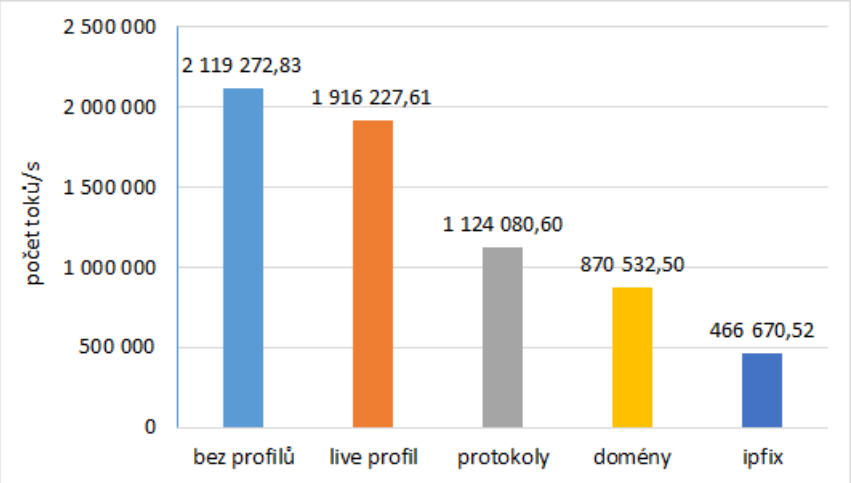
bez profilů - První scénář testuje kolektor bez zapnutého profilování. Navíc není použit žádný výstupní modul. Mohlo by totiž docházet ke zkreslení výsledků kvůli pomalému zápisu na disk. Tento scénář představuje maximální možnou efektivitu mediátoru bez jakékoli činnosti. Největší zpoždění tak představuje vstupní plugin a samotný TCP přenos.

live profil - Druhý test již zahrnuje profilování dat, avšak pouze v nejzákladnější podobě. To znamená, že je vytvořen jen live profil, do kterého jsou všechna data zařazena. Tento test ukazuje zejména paměťovou náročnost správy profilů a také výkonnostní rozdíl při jednom průchodu všech datových záznamů s takřka nulovou činností.

protokoly - Třetí test již obsahuje profilování dat dle aplikačních protokolů zmíněných výše. To znamená, že u každého datového záznamu je vyhledán zdrojový a cílový port a je porovnán s hodnotami ve všech filtračních pravidlech.

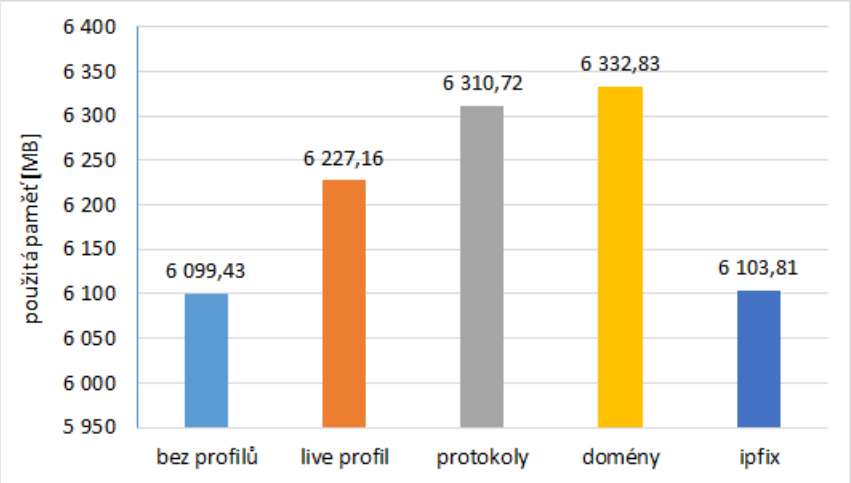
domény - Čtvrtý scénář zahrnuje celou hierarchii profilů. Tzn., že předchozí test je obohacen o podprofil s kanály dle jednotlivých domén v HTTP požadavcích. Toto filtrování spočívá v testování, obsahuje-li požadavek na konci své hodnoty řetězec z filtračního pravidla.

ipfix - Poslední test profilování dat neobsahuje. Místo něj je na výstupu zapojen jednoduchý modul, který data ukládá na disk v jejich binární podobě, tj. jako IPFIX pakety. Tento test je zde uveden pro porovnání, jak moc profilování ovlivní celkový chod kolektoru. Typicky se totiž jedná právě o výstupní moduly, které zpracování nejvíce zpomalují.

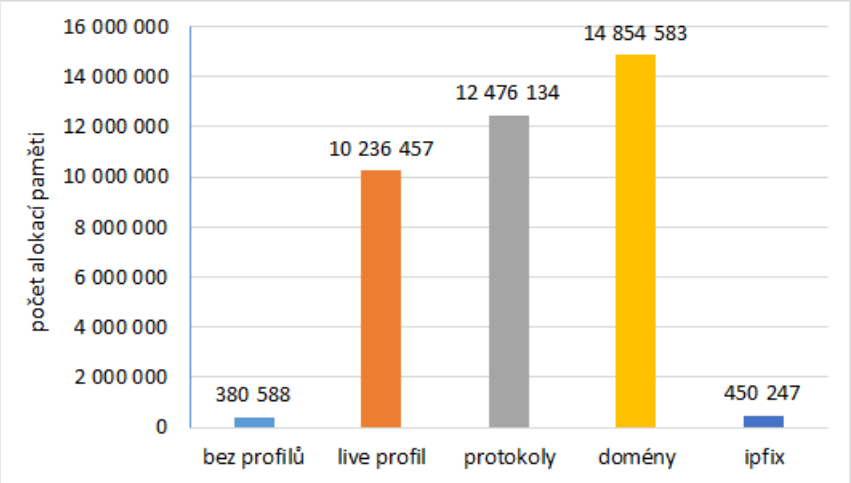


Obrázek 4.4: Rychlost zpracování dat

Nejprve jsem tedy testoval propustnost mediátoru. Data byla posílána po síti z jednoho zdroje pomocí TCP spojení. Výsledky testů jsou vyneseny do grafu na obrázku 4.4. Z grafu je zřejmé, že samotné zapojení profilování tvoří pouze minimální zátěž. Nicméně již při profilování dle zdrojových a cílových portů je vidět úbytek výkonu. Jedním z důvodů je vyhledávání elementů v datovém záznamu. Aby mohla být hodnota porovnána s filtračním pravidlem, musí se dle šablony sekvenčně projít celý datový záznam, dokud se nenalezne pozice, na které se tento prvek nachází. Tento problém je možné částečně eliminovat vytvořením pomocného seznam pozic některých prvků pro každou šablonu. Nejčastěji se pro filtrování používají právě porty a IP adresy. Při pokusné implementaci takového řešení došlo k navýšení průchodnosti o zhruba 10%. Dalším problémem je samotné procházení pole všech datových záznamů a porovnávání hodnot s filtračními pravidly. Toto se neprojevuje při live profilu, jelikož se neskáče do kódu pro filtrování hodnot a tak se minimalizuje počet volaných funkcí pro každý záznam. Problém sekvenčního zpracování by mohl být řešen paralelizací tohoto průchodu a profilování, jelikož jednotlivé iterace cyklu na sobě nijak nezávisí.



Obrázek 4.5: Paměťová náročnost mediátoru



Obrázek 4.6: Počet alokací

Při pohledu na graf na obrázku 4.6 lze pozorovat prudký nárůst počtu alokací paměti. To je následkem vytváření pole profilů pro každý datový záznam. Obrázek 4.5 dokazuje, že celkové využití paměti již tak rozdílné není. Jedná se tedy jen o několik bytů pro každý záznam. Pro snížení počtu alokací by bylo vhodné do mediátoru implementovat efektivnější správu paměti, aby nedocházelo k její neustálé alokaci a následnému uvolňování. Výsledky testovacího scénáře s doménami ukazují, že přidání úrovně profilování na celkovou výkonnost nemá příliš velký vliv. V tomto případě už nelze vyhledávání elementů v záznamech urychlit, jelikož prvky HTTP požadavků mají proměnnou délku. Nelze si tak zapamatovat jejich pozici a proto musí být v záznamu nalezeny vždy jeho sekvenčním průchodem. Při profilování byl kolektor stále schopen zpracovat o několik set tisíc toků za vteřinu více, než bez profilování a ukládání do IPFIX formátu.

Dalším testem bylo porovnání propustnosti profilovacího modulu s ostatními pluginy. Graf na obrázku 4.7 ukazuje rychlost zpracování mediačních modulů mediátoru. Testovány byly tyto moduly:

dummy - Jedná se o primitivní modul, který pouze vypíše hlášku o přijetí zprávy a předá ji k dalšímu zpracování.

anonymization - Realizuje anonymizaci IPv4 či IPv6 adres.

joinflows - Modul spojuje data od jednotlivých zdrojů dle ODID. To provádí změnou identifikátoru v hlavičce paketu a spojováním stejných šablon z různých zdrojů. Zároveň do každého datového záznamu přidává element s informací o původním identifikátoru.

filter - Jedná se o zjednodušenou verzi profilovacího modulu, oproti němu však postrádá některé důležité optimalizace.

odip - Přidává do každého datového záznamu informaci o IP adrese zdroje dat.

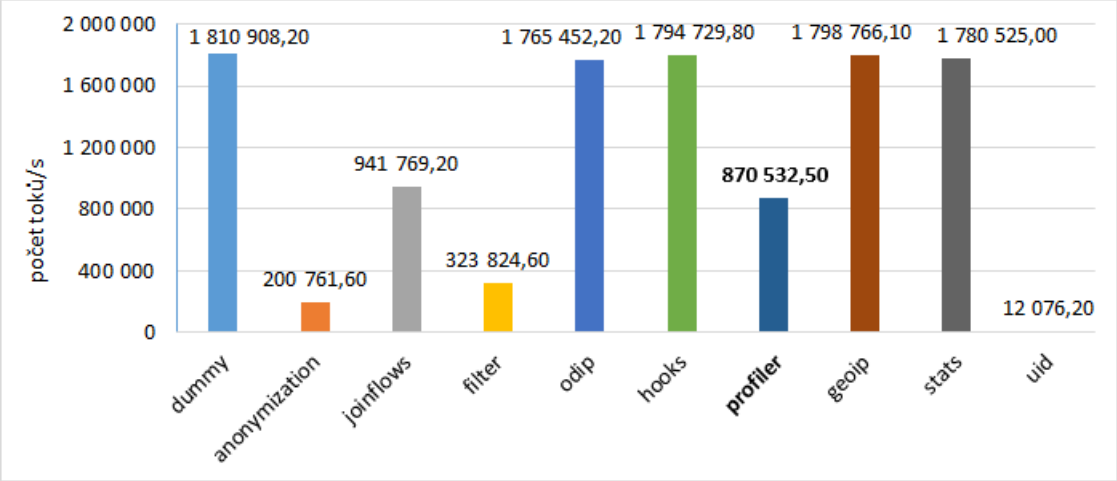
hooks - Modul, který je schopen spustit uživatelem zadaný příkaz při různých událostech. Mezi ně patří například připojení nového exportéru.

profiler - Profilovací modul nastaven na profilování dle aplikačních protokolů a domén v HTTP požadavcích.

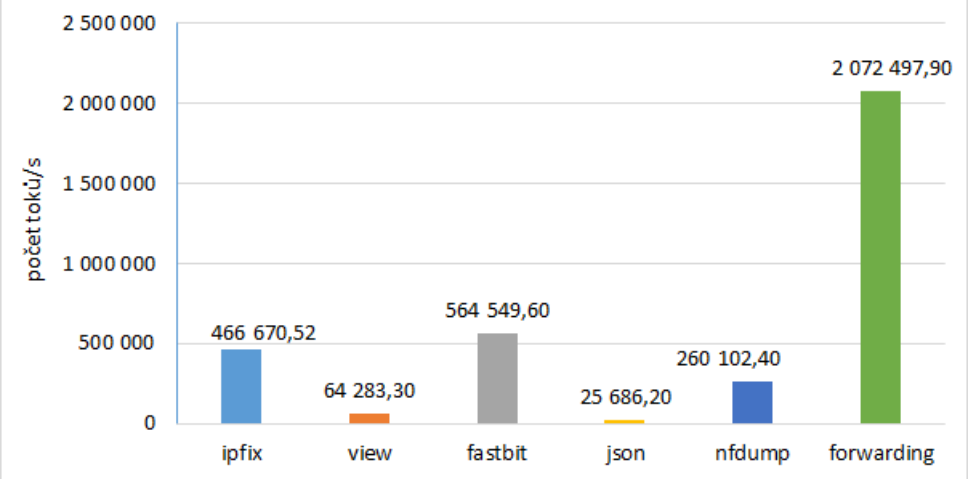
geoip - Geolokační modul, který z databáze získá kód země dle IP adresy.

stats - Počítá statistiky o tocích, bytech a paketech dle jednotlivých ODID.

uid - Plugin, který z SQL databáze získává různé údaje dle zdrojové a cílové adresy jednotlivých toků.



Obrázek 4.7: Rychlost mediačních modulů



Obrázek 4.8: Rychlost výstupních modulů

Obrázek 4.8 obsahuje poslední graf, znázorňující rychlost zpracování jednotlivých výstupních modulů. Do testu byly zahrnuty tyto pluginy:

ipfix - Ukládání dat ve formátu IPFIX, stejné jako v testu výkonnosti profilovacího modulu.

view - Data nejsou ukládána, ale přímo tištěna na standardní výstup.

fastbit - Modul, který pro ukládání dat využívá knihovnu FastBit[2].

[2]https://sdm.lbl.gov/fastbit/

json - Datové záznamy jsou převedeny do formátu JSON a odeslány po síti.

nfdump - Data jsou uložena na disk ve stejném formátu, jaký používá nástroj nfdump.

forwarding - Plugin přeposílající data po síti v IPFIX formátu.

Z grafů je zřejmé, že profilovací modul nepatří mezi nejrychlejší mediační moduly. To je způsobeno zejména jeho komplexností a nutností vyhledání a porovnání hodnot v každém datovém záznamu. I tak ale většinou nezpůsobuje zpomalení mediátoru, jelikož skoro všechny výstupní pluginy jsou výrazně pomalejší.

Kapitola 5

Závěr

Cílem této práce bylo seznámit se s problematikou monitorování sítí a sběru a ukládání dat na architektuře IPFIX. Po studiu této problematiky následoval návrh řešení pro profilování dat pomocí nástroje IPFIXcol. Hlavními požadavky bylo jednoduché vytváření profilů a zadávání filtračních pravidel, vysoká efektivita a nízká paměťová náročnost. Výsledné řešení splňuje specifikaci a vyhovuje případům použití.

Prvním krokem bylo nastudování si informací o monitorování sítí a o dostupných řešeních. První část práce pojednává o mnou zjištěných informacích z dostupných zdrojů. Cílem mého snažení bylo poskytnutí obecného náhledu do této problematiky a popis existujících architektur, protokolů a nástrojů, které se monitorováním sítí a sběrem dat zabývají.

Následně jsem se seznámil s IPFIX mediátorem. Naučil jsem se pracovat s nástrojem IPFIXcol. Pochopil jsem, na jakém principu funguje a jak je možné jeho funkcionalitu co nejlépe rozšířit a vylepšit. Zároveň jsem se seznámil s nástroji nfdump a NfSen, které jsou ekvivalentem IPFIXcolu na architektuře NetFlow.

Po seznámení se se všemi nástroji jsem na základě studia a analýzy informací navrhl řešení pro profilování dat. Můj návrh je založen na existujícím a osvědčeném řešení v nástroji NfSen, tj. na hierarchii profilů a kanálů, do kterých jsou data zařazována. To probíhá na základě filtračních pravidel. Každý kanál obsahuje právě jedno pravidlo. Profily jsou pak logickým zapouzdřením skupin kanálů.

Další fází byla implementace navrženého řešení. Celá konfigurace profilů a kanálů je uložena v XML souboru. Filtrační pravidla jednotlivých kanálů jsou zadávána s využitím textových názvů IPFIX elementů. Transformována jsou do stromové hierarchie. Při filtrování dat pak dochází k rekurzivnímu procházení tohoto stromu. Ačkoli je jádro mediátoru implementované v jazyce C, pro své řešení jsem použil jazyk C++. Ten jsem zvolil hlavně z důvodu využití standardních kontejnerů a možnosti zapouzdření dat do tříd. Část řešení, která je přímo v jádře mediátoru, je připojena jako statická knihovna obsahující rozhraní v jazyce C. Druhá část je ve formě mediačního pluginu, který je kolektorem dynamicky připojen za běhu.

Poslední část práce je věnována výsledkům kontroly správného profilování a měření výkonnosti. Z testu správnosti profilování vyplývá možnost neomezeného využití pro monitorování sítě, tvorbu statistik a detekci anomálií. V porovnání s výkonem kolektoru při klasickém provozu s ukládáním na disk nedochází ke snížení výkonu. Paměťová náročnost se zvyšuje pouze minimálně.

Práce otvírá další možnosti rozšíření IPFIX mediátoru. Je například možné navrhnout a implementovat externí nástroj, který by se o správu profilů staral a poskytoval rozhraní pro jeho ovládání. Bylo by tak možné toto napojit například na grafické uživatelské rozhraní

pro zobrazování, modifikaci, přidávání a rušení profilů a kanálů. Další možností pokračování práce je zefektivnění samotného profilování, pokud by stávající řešení přestalo být dostačující. Prostor je zejména v paralelizaci filtrování a efektivnější správa paměti. Díky funkčnímu profilování dat mohou vznikat i další moduly, které těchto informací využívají. Pro potřeby této práce tak vznikl mediační plugin, který zaznamenává statistické údaje právě dle přiřazených profilů. Díky tomu bylo možné rychle a efektivně zkontrolovat a vizualizovat funkčnost profilování.

Literatura

[1] Brownlee, N.; Mills, C.; Ruth, G.: Traffic Flow Measurement: Architecture. RFC 2722, RFC Editor, Říjen 1999.
URL https://tools.ietf.org/html/rfc2722

[2] Case, J. D.; Fedor, M.; Schoffstall, M. L.; aj.: A Simple Network Management Protocol (SNMP). RFC 1157, RFC Editor, Květen 1990.
URL http://tools.ietf.org/html/rfc1157

[3] Cesnet z.s.p.o.: IPFIXcol. https://github.com/CESNET/ipfixcol, Leden 2015.

[4] Cisco Systems: Understanding Simple Network Management Protocol (SNMP) Traps. Říjen 2006.
URL http://www.cisco.com/c/en/us/support/docs/ip/simple-network-management-protocol-snmp/7244-snmp-trap.html

[5] Claise, B.; Bryant, S.; Leinen, S.; aj.: Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. RFC 5101, RFC Editor, Leden 2008.
URL https://tools.ietf.org/html/rfc5101

[6] Claise, B.; Trammell, B.; Aitken, P.: Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. RFC 7011, RFC Editor, Září 2013.
URL https://tools.ietf.org/html/rfc7011

[7] Espa?ol, V. C.: *Network traffic classification : from theory to practice*. Diplomová práce, Universitat Polit?cnica De Catalunya. Departament D'Arquitectura De Computadors, 2014.

[8] Kobayashi, A.; Claise, B.; Muenz, G.; aj.: IP Flow Information Export (IPFIX) Mediation: Framework. RFC 6183, RFC Editor, Duben 2011.
URL https://tools.ietf.org/html/rfc6183

[9] Matoušek, P.: *Síťové aplikace a jejich architektura*. VUTIUM, Brno, 2014, ISBN 978-80-214-3766-1.

[10] McCloghrie, K.; Rose, M. T.: Management Information Base for Network Management of TCP/IP-based internets: MIB-II. RFC 1213, RFC Editor, Březen 1991.
URL https://tools.ietf.org/html/rfc1213

[11] Quittek, J.; Zseby, T.; Claise, B.; aj.: Requirements for IP Flow Information Export (IPFIX). RFC 3917, RFC Editor, Říjen 2004.
URL https://tools.ietf.org/html/rfc3917

[12] Sadasivan, G.; Brownlee, N.; Claise, B.; aj.: Architecture for IP Flow Information Export. RFC 5470, RFC Editor, Březen 2009.
URL https://tools.ietf.org/html/rfc5470

[13] Sadasivan, G.; Valluri, V.; Djernaes, M.: Cisco Systems NetFlow Services Export Version 9. RFC 3954, RFC Editor, Říjen 2004.
URL http://tools.ietf.org/html/rfc3954

[14] Trammell, B.; Boschi, E.: An Introduction to IP Flow Information Export (IPFIX). *IEEE Communications Magazine*, ročník 49, č. 4, Duben 2011: s. 89 – 95.

[15] Trammell, B.; Boschi, E.; Mark, L.; aj.: Specification of the IP Flow Information Export (IPFIX) File Format. RFC 5655, RFC Editor, Říjen 2009.
URL https://tools.ietf.org/html/rfc5655

Příloha A

Obsah CD

Na CD jsou umístěny zdrojové soubory nástroje IPFIXcol. V odděleném adresáři jsou ty, které přímo souvisí s tématem této práce. Disk dále obsahuje soubor README a tuto práci včetně zdrojových textů.